Coursework Report

Niko Triantafillou

40405829@napier.ac.uk

Edinburgh Napier University – Advanced Web Tech (SET09103)

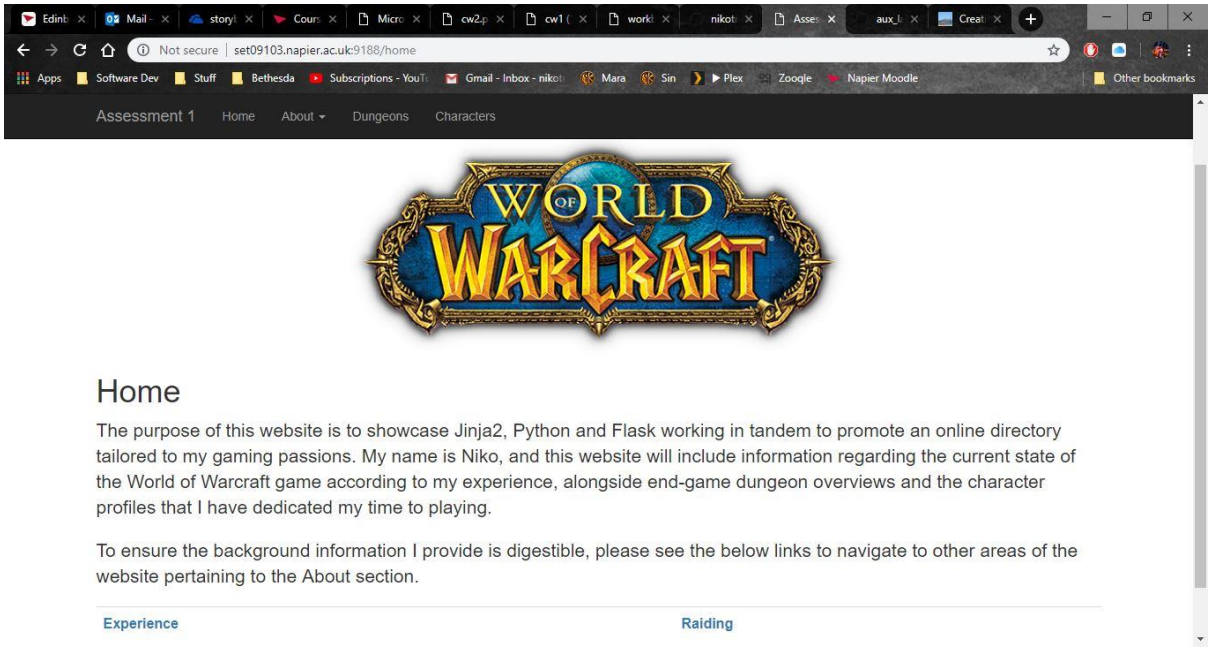
Assessment 1: World of Warcraft

Introduction

Assessment 1: World of Warcraft is a web application that utilises the technologies of Python, Flask and Jinja2 to create a dictionary pertaining to the current end-game content of the latest titular Battle for Azeroth expansion, as well as some personal background information revolving the author: Niko Triantafillou.

The web application takes advantage of Flask routing to ensure the traversal of the application is as smooth as possible for the user, whilst operating Flask static files for hosting the Bootstrap framework in addition to the various images used throughout the application.

The specific content found in the web application orbit around the Player versus Environment (PVE) aspect of the game, such as dungeon and raid group content, in addition to the various playable character races and classes within the game. The online compendium gives accurate in-game descriptions of the playable found in this web application dictionary, and links it all together using breadcrumbs so that the user has a clear understanding of where they are during navigation of the application.

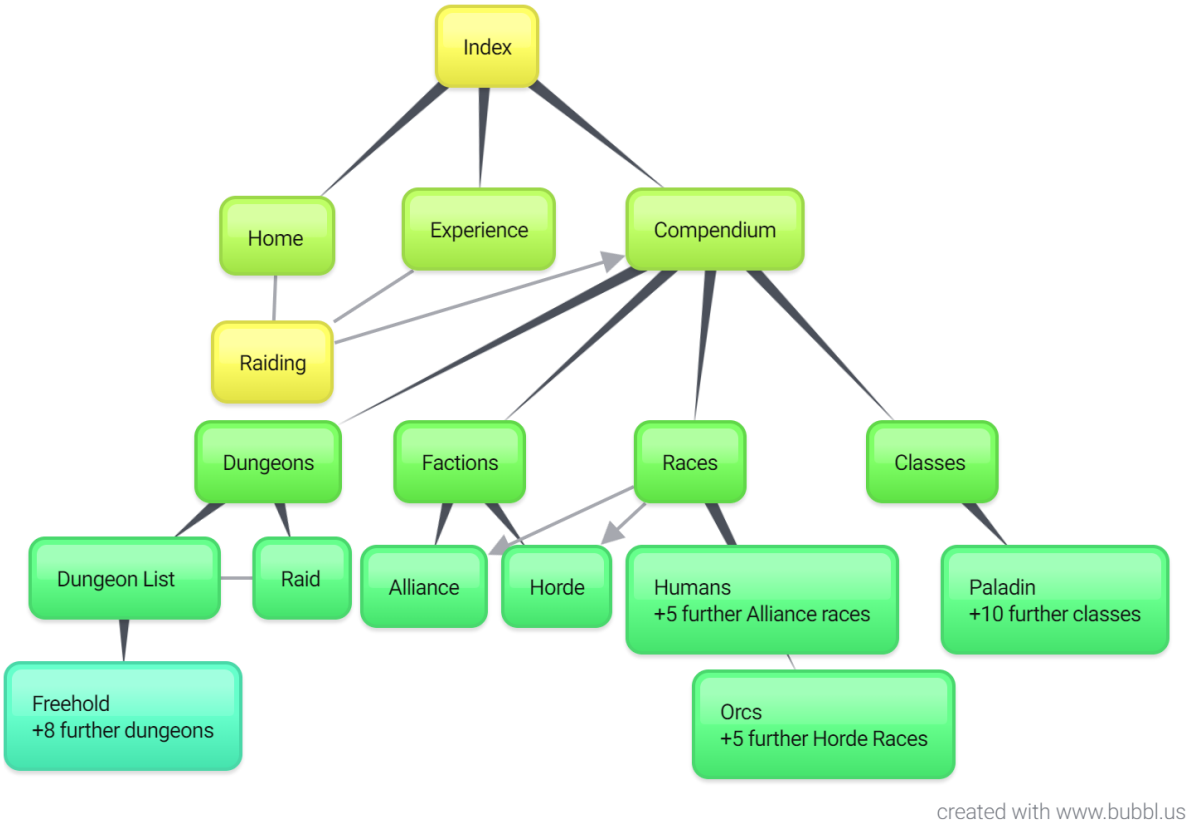


Design

The layout is a basic template layout using Bootstrap version 4.1.3, straight-out-of-the-box. Outside of the image placements, sourced from various Google hit results, the navigation bar has been customised with working and personalised links that help the user navigate throughout the web application.

Furthermore, I have taken advantage of the breadcrumb Bootstrap component to help keep the user informed of their journey throughout the application. This ultimately aids in user accessibility and familiarity, and creates a nice, content driven network of relevant information and hyperlinks.

For the body content, I have used the Bootstrap container wrapper and aligned images using either text-alignment tags or <img> class containers, depending on what was required in my code base at that time, and what I was trying to achieve with the placement of the images used.



Enhancements

The features that I would add would be an interactive character creator based on the existing modules and data; this would allow the user to dynamically pick and choose based on faction, race and class what they would like to play as if they had access to the World of Warcraft game. I would also look to improve on the existing data structure of the aforementioned content, linking them together on a deeper level so that users can navigate between each class and also view the specific races that are available to them.

Critical Evaluation

I have structured the architecture so that every file has a proper namespace and location. From the top-level folder, the "run.py" file can be utilised in Linux to launch the application. Within the "source" folder, the "__init__.py" file creates an instance of the Flask module, whilst the "views.py" file controls the Flask routes and Jinja2 is used to import the Bootstrap CSS and JS files into the application from the "header.html" and "footer.html" files respectively.

All the content found throughout the application uses the same ruleset outlined above, the race, class and dungeon lists are ordered into their respective folders and are kept separate from the base templates, and I feel this works well as it ensures the structure directory is clean and organised.

Personal Evaluation

I feel like I performed quite well. Using the available resources throughout the module workbook and additional resources found on the Internet, I was able to create a simple web application dictionary for a subject I am passionate about.

Using my personal experience in software development (I am a self-taught VB/C# .NET junior developer with 2 years of experience in the industry), I was able to quickly understand the way Python, Flask and Jinja2 were presented and make connections on how these technologies worked on a fundamental level.

The challenges I faced were specifically tailored to learning the Linux operating system; I found it incredibly difficult using a command-line based system for the entirety of the applications development, but I eventually found ways to circumvent or make things easier when re-iterating testing cycles of the application (the Up arrow is your best friend to repeat past commands).

References

- To gain a better understand of directory structuring: https://www.patricksoftwareblog.com/creating-a-simple-flask-web-application/?fbclid=IwAR2P0_U1vHtAtWy-PE5h94K8SE7qdO8ADx_jGOjNUvnSlo5y5vOe7RlcrFQ
- To understand Linux better: https://www.stackoverflow.com
- To understand Python Flask better: https://www.stackoverflow.com
- For Bootstrap development: https://getbootstrap.com/docs/
- For images: https://www.google.com
- For Navigation Map: https://bubbl.us